Real-time Monocular 6DoF Tracking of Textureless Objects using Photometrically-enhanced Edges

Lucas Valença[1][a], Luca Silva[1][b], Thiago Chaves[1], Arlindo Gomes[1][c], Lucas Figueiredo[1][d], Lucio Cossio[2], Sebastien Tandel[2], João Paulo Lima[3,1], Francisco Simões[4,1][e] and Veronica Teichrieb[1][f]

[1]*Voxar Labs, Centro de Informática, Universidade Federal de Pernambuco, Recife, Brazil*

[2]*HP Inc., Porto Alegre, Brazil*

[3]*Departamento de Computação, Universidade Federal Rural de Pernambuco, Recife, Brazil*

[4]*Campus Belo Jardim, Instituto Federal de Pernambuco, Belo Jardim, Brazil*

Keywords: 6DoF Tracking, Edge-based, Monocular, Real-time, Model-based, Mobile, Multi-object, Augmented Reality, Textureless, Segmentation, RGB, HSV, Single-thread.

Abstract: We propose a novel real-time edge-based 6DoF tracking approach for 3D rigid objects requiring just a monocular RGB camera and a CAD model with material information. The technique is aimed at low-texture or textureless, pigmented objects. It works even when under strong illumination, motion, and occlusion challenges. We show how preprocessing the model's texture can improve tracking and apply region-based ideas like localized segmentation to improve the edge-based pipeline. This way, our technique is able to find model edges even under fast motion and in front of high-gradient backgrounds. Our implementation runs on desktop and mobile. It only requires one CPU thread per object tracked simultaneously and requires no GPU. It showcases a drastically reduced memory footprint when compared to the state of the art. To show how our technique contributes to the state of the art, we perform comparisons using two publicly available benchmarks.

1 INTRODUCTION

Tracking the 6DoF pose of a 3D object through monocular videos in real-time has been a long-time computer vision challenge. Multiple tasks in thriving technological fields (e.g., extended reality, robotics, autonomous vehicles, medical imaging) require information regarding the objects populating the environment. Real-time tracking requires high precision with temporal consistency at every snapshot of the scene. In addition, dynamic conditions of the real world may impose numerous difficulties, such as unexpected occlusions, foreground or background clutter, varying illumination, and motion blur. Hardware challenges are also present (e.g., lack of processing power or sufficiently reliable scene input from available sensors).

It is also important to briefly mention detection techniques, which are often seen as complementary to tracking works. The main difference between them lies in the temporal consistency and computational cost. Usually, the detection step (which can be automated or even manually aligned) provides an initial estimation for trackers to use. Unlike detectors, trackers achieve higher efficiency by not needing to search the entire frame every time, as the object is expected to be in the vicinity of where it was in the last frame. Additionally, trackers try to make the current frame's estimation coherent with respect to the previous frames, generating smoother results.

Objects being tracked can vary in appearance, being considered textured if there is a significant amount of visual information such as internal gradients and color variation. The lack of such information in textureless objects configures a relevant category considering how widely common these objects are (e.g., 3D prints, industrial parts), and how suitable their textureless surfaces are for applications such as augmented reality.

[a] https://orcid.org/0000-0003-2625-9857
[b] https://orcid.org/0000-0003-2182-9208
[c] https://orcid.org/0000-0003-3930-2754
[d] https://orcid.org/0000-0001-9848-5883
[e] https://orcid.org/0000-0001-9368-2298
[f] https://orcid.org/0000-0003-4685-3634

Valença, L., Silva, L., Chaves, T., Gomes, A., Figueiredo, L., Cossio, L., Tandel, S., Lima, J., Simões, F. and Teichrieb, V.
Real-time Monocular 6DoF Tracking of Textureless Objects using Photometrically-enhanced Edges.
DOI: 10.5220/0010348707630773
In *Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2021) - Volume 5: VISAPP*, pages 763-773
ISBN: 978-989-758-488-6

This paper presents a novel 6DoF object tracking technique (see Fig. 1) based on edges and photometric properties. Our technique is mainly aimed at rigid, textureless or low-texture objects with pigmentation (see Section 3.4). It uses previously known material information from the input 3D model to generate photometrically-robust local color data. This data undergoes statistical simplification in order to generalize it, enabling matching to real-world objects. During tracking, the color matching process then generates a binary segmentation mask in real time. We demonstrate that our approach is capable of matching object edges even under motion blur or in front of high-gradient backgrounds, helping to overcome what is arguably the largest pitfall of edge-based tracking. We also introduce a novel, more computationally efficient way to match object's 2D and 3D silhouette points and a new search range concept (Valença, 2020).

In terms of hardware, the only required input sensor is a monocular RGB camera, such as a usual mid-range commodity webcam, and a single CPU core per object. The proposed technique is also, to the best of our knowledge, the first monocular real-time 6DoF object tracker among state-of-the-art contenders to have been shown to work in mobile devices. For this reason, we argue that the proposed work is more suitable for lightweight application scenarios than currently existing methods. Additionally, most fields that make use of object tracking commonly utilize it as part of a larger pipeline (running simultaneously with other algorithms). Thus, sparing resources by just using one core of the (usually multi-core) modern CPU can be considered a desirable feat.

Our work's main contributions are summarized as:

- A novel multi-platform edge-based approach aimed at textureless, pigmented objects which provides comparable precision to the state of the art with less hardware requirements, lower runtime, and a memory footprint orders of magnitude smaller. The technique excels in handling illumination changes and fast motion. It also contains a new way for edge-based tracking to isolate object edges under blur and high-gradient backgrounds.
- Sequences using 3D printed textureless objects.

2 RELATED WORK

Due to the various tracking challenges that objects and scenes can present, very distinct methods exist in the literature. Feature and SLAM-based approaches, which use feature descriptors to extract object and scene characteristics, have been adapted to 6DoF detection and tracking. These are particularly popular in industry applications for mobile devices[1], generally coupled with 3D reconstruction (not model-based). In the academia, ORB-SLAM2 (Mur-Artal and Tardós, 2017) has been adapted to track 3D rigid objects (Wu et al., 2017) in a benchmark scenario. These methods tend to struggle with textureless objects and perform best in unchanging scenes with fixed illumination and object position (relative to the scene).

Recently, monocular region-based approaches have been proposed (Tjaden et al., 2018; Zhong and Zhang, 2019; Stoiber et al., 2020), and most famously PWP3D (Prisacariu and Reid, 2012). These techniques track a whole portion of the input frame and statistically differentiate between object and background colors to isolate a silhouette and estimate the motion. One such method, RBOT (Tjaden et al., 2018), can be seen as an evolution of PWP3D and is, to our knowledge, the approach that has so far been proven to yield the most reliable results in multiple versatile conditions. It is suitable to a wide range of objects and textures, and has been thoroughly evaluated in multiple benchmarks. Very recently, The main problem with real-time region-based approaches lies in their computational cost (e.g., PWP3D relies heavily on GPGPU, and RBOT on multi-threading and GPU rendering). This parallelism bottleneck comes to light especially for multi-object tracking. Very recently, a work has proposed a sparse region-based approach that might help overcome this performance issue (Stoiber et al., 2020).

Edge-based approaches have been around the longest and tend to be more lightweight (Seo et al., 2013). They attempt to match the model's edges to gradients found in the frame. The main edge-based pipeline was popularized by RAPID (Harris and Stennett, 1990) and a wide array of trackers (Han and Zhao, 2019) have been developed on top of the RAPID pipeline, including the proposed method. Initially, RAPID-based works relied only on differences of grayscale intensity. Then, techniques like GOS (Wang et al., 2015) started applying color analysis to regions neighboring the edges. Edge-based techniques tend to struggle with highly textured backgrounds, as the many gradients can cause erroneous matching. Additionally, they are prone to fail in cases of heavy motion where edges become blurred. Thus, since the appearance of recent region-based approaches like RBOT, to our knowledge, edge-based works have fallen out of the state of the art (with one exception, which is not real-time capable) (Bugaev et al., 2018).

For completeness sake, we shall also mention a relevant RGB-D deep learning approach (Garon and

[1] Apple ARKit

Figure 1: Our work in different scenes. Left to right: flashing lights, occlusion, clipping, motion blur, multi-object, and AR.

Lalonde, 2017). It has not achieved the same runtime speed of machine learning RGB-D trackers (Tan et al., 2017). Still, it was shown to be reliable in scenarios with strong occlusions or imprecise initialization. As downsides, we must mention the hardware requirements (RGB-D cameras and high-end GPUs).

3 METHOD

Our method uses the object's 3D model together with its material information. Models are defined as standard triangular meshes composed by a set of vertices, edges, and faces, without repetition of shared edges between faces. Materials can be assigned to the entire object or to a subset of faces. Diffuse texture images are assumed to be in RGB format with 8-bit quantization per intensity channel. The color of faces with no diffuse texture specified in its material (as is the case with textureless objects such as solid-color 3D prints) is referred to as the RGB vector $C \in \mathbb{R}^3 \rightarrow \{0,\cdots,255\}^3$. Such color is specified by the diffuse color coefficient of the face's material.

All frames are remapped to remove radial and tangential lens distortion. We also assume the camera to be precalibrated, with the focal length and principal point expressed in pixels as the 3×3 intrinsic parameter matrix K, as per the pinhole camera model.

The extrinsic parameter matrix E describes the 6DoF transform from world to camera space and is represented as a 3×4 matrix $[R \;\; t]$. The projection $p \in \mathbb{R}^2$ of an object point $P \in \mathbb{R}^3$ can be described as $p = \beta(K \cdot (E \cdot \tilde{P}))$, with $\tilde{P}$ being its homogeneous coordinate form. Additionally, β represents the conversion between $\mathbb{P}^3 \rightarrow \mathbb{R}^2$ so that $\beta(P) = [X/Z, Y/Z]^T \in \mathbb{R}^2$.

The pose E_i for the i^{th} frame can be estimated by approximating the object's infinitesimal rigid body motion between frames. This is done by using the i^{th} frame and the last known pose expressed by $E_{(i-1)}$ to estimate $\Delta t = t_i - t_{(i-1)}$ and ΔR, finally obtaining E_i.

3.1 Model Preprocessing

Before tracking begins, the 3D model is preprocessed and has its texture simplified. This is a one-time procedure. Every model face's diffuse texture image region (i.e., the diffuse image associated to the set of vertices, masked by the UV coordinates associated with each face vertex) is simplified by being quantized into a 32-bin RGB probability histogram. Each color's probability value corresponds to the percentage of pixels of said color in the face's histogram. Colors are then ranked according to said probability, highest first, and stored in HSV format. They are stored until either reaching a $k_{p1} = 95\%$ accumulated probability limit (i.e., all stored colors for that face represent over $k_{p1}\%$ of the face's surface) or reaching the $k_{p2} = 3$ color per face limit. For faces without diffuse texture, $C = 100\%$ probability. The k_{p1} limit was set to avoid storing unrepresentative colors and k_{p2} was set due to our textureless scope, as an upper limit for the amount of colors and gradients per triangle.

Each face color is then assigned to every edge within the face. Thus, every edge has a list of all colors from all faces that contain it, without repetition. The list of edges, vertices that form them, and HSV colors they contain are then stored. The original object model and materials are no longer utilized.

3.2 Efficient Silhouette Encoding

The first step executed in real-time for every frame consists of encoding the 2D-3D correspondences for the current pose. All the model's 3D vertices are projected using the last known pose. Based on line projection properties, the 2D edges are drawn using the projected vertices and Bresenham's line-drawing algorithm. The canvas used consists of a single-channel 32-bit image. Each pixel of a projected edge has its intensity value corresponding to its own 3D edge index from the edge set ξ contained in the preprocessed model. This is similar to the Edge-ID algorithm (Lima et al., 2009), but uses a single 32-bit channel directly for the ID instead of splitting it in three 8-bit channels. The resulting image is referred as the *codex*. It is important to mention that our implementation does not utilize depth testing (for details, see the Appendix).

Next, a topological border following algorithm (Suzuki and Abe, 1985) is used on the codex to extract the external-most 2D contour ζ. The contour is continuous and returned in counter-clockwise

order. This set of pixels corresponds to the object's 2D silhouette at the last known 6DoF pose.

Finally, the set of 2D normal vectors Υ is determined so that $\Upsilon_i = \tilde{\Theta}(\zeta_{(i+1)} - \zeta_{(i-1)})$, as ζ is circular and continuous. The vector $\tilde{\Theta}$ corresponds to the normalized output of the function $\Theta(v) = [-y,x]^T$, which returns a 2D vector orthogonal to $v = [x,y]^T \in \mathbb{R}^2$. The normal vector is always rotated 90^o clockwise. Given the counter-clockwise property of the contour's continuity, this means the 2D normals will automatically always point outwards from the silhouette.

To extract a 3D point P from the codex's 2D point p, because we only projected the edge vertices, we need to linearly interpolate those two endpoint Z values to obtain P_z. Then, $P = [(p_x - c_x)/(P_z \cdot f), (p_y - c_y)/(P_z \cdot f), P_z]^T$ (Lima, 2014), where c_x, c_y, and f come from the intrinsic camera matrix K defined previously. To find those two endpoint Z values we see the edge ID in the codex and verify in the edge array which pair of 3D vertices compose that edge. This operation is only performed when a match is found for that silhouette pixel, which is more efficient than calculating all Z values using a z-buffer.

3.3 Adaptive Circular Search

In order to segment the frame, we need to establish a search range. Instead of conducting 1D linear searches directly (as is traditional in edge-based approaches), our search is performed locally in dense circular areas with radius r, centered around contour points from ζ (drawing inspiration from RBOT). The goal in our case, though, is to match the colors found in the frame with the ones expected from that region of the preprocessed 3D model. Additionally, instead of sampling 3D points at random, we're taking consecutive points directly from ζ in 2D. This is done to maintain uniform contour coverage as the projected silhouette's length varies with z-axis translation.

Ideally, every point in ζ should be evaluated in its own circle, covering a dense strip around the object's curvature. Because that is too computationally expensive, we are instead sampling every r^{th} consecutive contour point so that every circle is always equally re-visited by each of its neighbors, increasing coverage. Neighbors checking the circle area can be useful when there's larger motion between frames or colors are blended due to blur.

Instead of using a fixed radius value (as is commonly done in tracking approaches), we suggest to change the radius according to the object's current silhouette area in relation to the frame resolution. This way, r is updated at every new pose, proportionally to how many pixels are being occupied by the object's projection. This region consists of all pixels inside (and including) ζ, referred to as $A(\zeta)$. Thus, the amount of pixels inside this region is represented by $|A(\zeta)|$. The radius r can be calculated as a linear function with limiting thresholds:

$$r(\zeta) = min\left(\left\lfloor\left(k_\theta \cdot \lambda_A^{-1} \cdot |A(\zeta)| + r_0\right) \cdot \lambda_S\right\rfloor, r_1\right), \quad (1)$$

where $r_0 = 7$ pixels is the minimum tolerable level of detail where shapes are still discernible for our PnP-based approach, determined empirically. The angular coefficient $k_\theta = 0.002$ determines how fast the radius grows with proportion to the object size. The value of the maximum allowed radius is defined as $r_1 = \sqrt{k_A \cdot |A(F)|/\pi}$, where $|A(F)|$ is the total area (in pixels) of the input frame F and $k_A = 0.01$, or 1%, corresponds to the percentage of the frame that the maximum radius is allowed to occupy. The suggested values for k_θ and k_A were found via experimentation with multiple handheld objects.

Variable $\lambda_A = |A(F)|/(R_w \cdot R_h)$ represents the scale factor between the current resolution and R, the *base resolution* (in our case VGA). It makes it so our technique behaves uniformly under resolution changes. Subscripts h and w represent height and width, respectively.

Finally, $\lambda_S = min(F_w, F_h)/min(R_w, R_h)$ represents the shortest side scale. It minimizes changes when under different (e.g., wider or narrower) aspect ratios. An alternative to using λ_S to handle changes in aspect ratio is to use directly-proportional elliptical search circles, but that would divert from the circular search concept by creating directional search bias.

3.4 Binary Segmentation

We construct a segmentation mask by evaluating every pixel within the search circles described previously. In an ideal world, where the model texture corresponds exactly to what we will find in the frame, the binary segmentation is very straightforward. Yet, due to the *reality gap*, we must establish tolerances for a pixel's color to be considered a match. These tolerances are controlled by what we call the *uncertainty coefficient*, or σ. The idea is that in a scenario where we're very sure the texture is accurate, such as in 3D printed objects or synthetically-rendered scenes and applications, the uncertainty (and therefore tolerance to variation) is low. But in a real world scenario, that tends to be larger (see Section 6.1).

The core idea of the proposed segmentation approach consists of utilizing the photometric properties of the HSV space to help in overcoming reality gap challenges and filter pixels. By utilizing hue and saturation information, we can more easily identify a

preprocessed color in cases where, if using a space like RGB, it might not be so clear. Because we rely on HSV, our approach focuses on pigmented objects. Grayscale (non-pigmented) objects do not possess reliable hue information (see the Appendix).

To take advantage of the HSV properties, we borrow a concept from region-based techniques: the smoothed unit step function (see Fig. 2). In our case, we want to statistically highlight the difference between regions of HSV space itself, as to continuously regulate the uncertainty tolerance levels proportionally to the amount of pigmentation present.

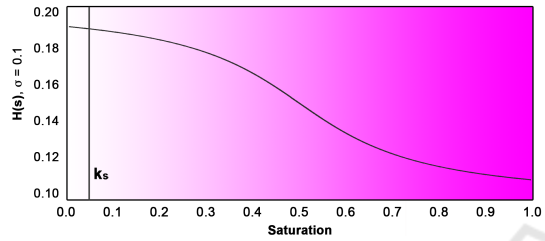

Figure 2: Behavior of $H(s)$ for $\sigma = 0.1$ as saturation changes. Desaturation threshold k_s is also highlighted.

This way, a segmented frame pixel p must satisfy the following conditions with respect to the preprocessed color c in order to be set as 1, or true, in the binary segmentation:

$$H(p_s) \geq d_\theta(p_h - c_h) \wedge p_s > k_s \wedge \eta \geq |p_s - c_s|, \quad (2)$$

where subscript h and s represent the hue and saturation channels, respectively. Function d_θ represents the shortest distance between hue values (considering they are cyclic) and constant $k_s = 0.05$ represents the minimum saturation threshold, indicating the point where there is so little pigmentation hue is no longer reliable (see supplementary material). The saturation tolerance range is mapped by the logarithmic function $\eta = k_l \cdot ln(\sigma) + k_\eta$, where the vertical shift $k_\eta = 0.5$ sets the log curve's plateau at the maximum saturation tolerance window possible, and the elasticity coefficient $k_l = 0.08$ both determines how fast that plateau will be reached and sets the function within the 0 to 1 quantization. Finally, H represents the hue tolerance proportional to the amount of pigmentation present, represented by a smoothed unit step function (Wang and Qian, 2018), adapted as follows:

$$H(s) = \sigma \cdot \left[\frac{3}{2} - \frac{1}{\pi} \cdot tan^{-1}\left(\frac{s - 0.5}{k_H}\right)\right], \quad (3)$$

where $k_H = 0.2$ represents the sharpness coefficient (Wang and Qian, 2018).

3.5 Mask Simplification

After every pixel in the search circles is subjected to the binary segmentation described in the previous section, we perform an additional check to avoid noise and erroneous background bits in the resulting mask. First, we identify all existing external-most contours (Suzuki and Abe, 1985) and store the largest contour in terms of area, assumed to be the main tracked object body (or body portion). Every contour with area less than $k_m = 5\%$ of the largest contour's is removed (see Fig. 3). This heuristic does not work if the erroneous portion is connected to the main body.

From this point, we will refer to the binary mask as Φ, so that $\{\Phi_f, \Phi_b\} \subseteq \Phi$ refer to the foreground (true) and background (false) pixel subsets.

Figure 3: Mask simplification process. From left to right: input frame, initial segmentation, final simplified result.

3.6 Correspondence Search and Pose Estimation

The next step consists of searching for a match for every point in ζ. We do this using a traditional (Drummond and Cipolla, 2002; Seo et al., 2013) edge-based linear 1D search for gradients along the normal vectors from Υ. This search uses the kernel $\psi = \begin{bmatrix}1 & -1\end{bmatrix}^T$ along the parametric pixel line described as $\zeta_i + dt\Upsilon_i$, and can be expressed as

$$S(i,t) = \psi \cdot \begin{bmatrix}\Phi_{\zeta_i + dt\Upsilon_i} & \Phi_{\zeta_i + d(t+1)\Upsilon_i}\end{bmatrix}, \quad (4)$$

where $d \in \{-1, 1\}$ corresponds to the search direction (inwards or outwards) so that $d = 1$ always points outwards from the model contour. The parametric term $t \in \{0, r-2\}$ is limited to $r-2$ to avoid matching with the edges of the circular search masks, as this indicates that the accepted pixel region would continue beyond the search range and does not correspond to a border. These cases are usually indicative of same-color background regions. The minimum range of 0 deals with cases where the object did not move at all.

Eq. (4) is calculated for every value of t until a result different from 0 is obtained. When and if that happens, the 2D-3D point tuple $(\mu_i,\ \zeta_i + dt\Upsilon_i)$ is stored in the vector χ as a possible match candidate. The 3D point μ_i represents the unprojected 3D equivalent of ζ_i from the codex.

Instead of alternating d values while t increases (Seo et al., 2013), we guide the direction of

d as per Table 1. Set $W = A(\zeta) - R(\zeta)$, where $R(\zeta)$ is a mask indicating the search range, where all pixels for every search circle for every chosen point in ζ are set to 1. Thus, $O = W(\zeta) \cap \Phi_f$ represents whether there seems to be an occlusion in Φ_f. This is due to no pixels in W being reachable by the search circles, but being within the contour, so that they can only be set if the contour was not interrupted by occlusion and could thus be filled, as per Fig. 4.

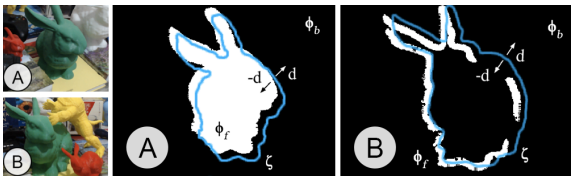

Figure 4: Segmentation without (A) and with (B) occlusion.

If the previous pose contour point $\zeta_i \in \Phi_f$, then the search continues outwards ($d = 1$) otherwise, if ζ_i is outside Φ_f, the search goes inwards ($d = -1$). If Φ_f is not filled and ζ_i is not in it, then it can be either outside or inside the current pose. Thus, we first assume it is inside the model and, to avoid matching the search strip, the first match going outwards is ignored, so that only the second one is valid. If no two matches are found, then the point is likely outside the model and we must search for the first match inwards.

Table 1: Guidelines for directional search and matching.

$\zeta_i \in \Phi_f$	$O = \varnothing$	d	**match stored in χ**
T	✖	1	1^{st}
F	F	-1	1^{st}
F	T	1 then -1	2^{nd} then 1^{st}

We would like to argue that this segmentation step is an interesting addition to be considered for edge-based approaches because it circumvents the background gradients problem by postponing the gradient matching step to the binary mask. The downside remains erroneous color-based segmentation, which can be adjusted according to the limitations of the chosen segmentation module for each technique.

Given a set of 2D-3D correspondences χ, the current pose is estimated as the solution of a PnP problem using a Levenberg-Marquardt optimization scheme with the last known pose as initial estimation.

4 IMPLEMENTATION DETAILS

This section provides details of our C++ implementation. All major steps are performed using a single CPU thread. The implementation relies on OpenCV, which is optimized for computational efficiency and multi-platform portability. Our mobile port utilizes the same C++ code via Android's NDK.

Every new input frame is converted to HSV and iterated by each tracker instance 12 times. Every iteration performs steps from Sections 3.2 through 3.6.

4.1 Tracking Multiple Objects

The technique has been extended to track multiple simultaneous objects, without compromising the frame rate, by having each tracker instance (one object per instance) running on a separate CPU thread. The amount of objects that can be tracked in parallel depends on the number of physical CPU cores.

Threads are managed in a producer-consumer scheme with the same number of threads and tracker-object instances, no instance being tied to any specific thread. One tracking job is queued per instance at every new frame, but are only processed after all jobs on the previous frame have finished, making it so the multi-object tracking is as fast as the slowest tracker instance. The approach is not very suitable to mobile CPUs, as cores do not usually share the same clock speed. Thus, our Android port does not currently support multi-object tracking.

5 3DPO Sequences

We made available our real-world qualitative sequences which we have named the 3DPO dataset (3D Printed Object dataset) together with CAD models and camera calibrations[2]. The set has scenes with multiple high-quality 3D-printed objects for the challenges of clutter, extreme motion, heavy occlusion, handling, and independent object-camera motion.

6 EVALUATION

Experiments were executed in a laptop with a dual-core Intel Core i7 7567U CPU @ 3.50 GHz, 16 GB of RAM, and integrated Iris 650 graphics. Results are compared to the current state of the art using the publicly available implementation of RBOT[3] (see the Appendix for details). We have used a reality-gap uncertainty $\sigma = 0.1$ on OPT and $\sigma = 0.01$ on RBOT dataset, unless where explicitly specified.

[2] github.com/voxarlabs/3DPO-Dataset
[3] github.com/henningtjaden/RBOT

6.1 Benchmark Comparison

For benchmark comparisons we used the publicly available datasets OPT[4] at 1920×1080 resolution and RBOT dataset[5] in its original resolution (640×512). Experiments were conducted using the area under curve (AUC) measure as described by OPT(Wu et al., 2017). Results can be seen in Table 2 and Fig.5.

Table 2: AUC score table for the OPT dataset.

Method	Soda	Chest	House	Ironman	Bike	Jet	Average
Proposed	4.11	**11.76**	**13.45**	**15.70**	0.04	0.94	**7.67**
RBOT	**6.62**	9.41	6.84	7.36	**6.44**	**8.74**	7.57

AUC results from Table 2 show that Chest, House, and Ironman were reliably tracked, as they contain the least non-pigmented parts and are less textured (better fitting our scope). One interesting aspect of our edge-based binary approach is that it is able to more finely match the object's silhouette than probabilistic approaches like region-based (see Fig.5).

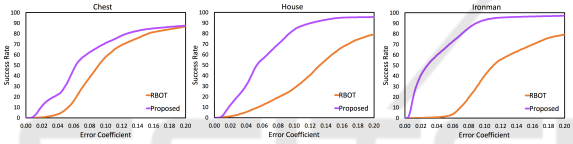

Figure 5: AUC curves for Chest, House, and Ironman.

Our method particularly struggled with objects Soda, Bike, and Jet. That was expected, as these objects are highly textured. In addition, Bike and Jet contain many non-pigmented portions and Soda has a highly symmetrical silhouette. In the case of Bike (Bugaev et al., 2018) and Jet, models and textures were also not very accurate. Thus, it is safe to say that these three objects presented a worse-case scenario for our work (see Fig.6).

Figure 6: Failure case which shows the proposed work attaching itself to the pigmented portions the Bike object.

Table 3 shows more detailed results per OPT challenge. There it can be seen that the proposed work particularly excels in handling illumination changes. This is mostly due to our segmentation utilizing hue information. In terms of handling motion, our technique's ability to swiftly recover from partial loss has been beneficial (see Section 6.2).

[4]media.ee.ntu.edu.tw/research/OPT/

[5]cvmr.info/research/RBOT/

Table 3: AUC scores per challenge.

Object	Method	Illum. Changes	Free Motion	Scale Changes	Translation	Rotation
Chest	Proposed	**8.75**	**9.77**	**13.69**	10.19	**12.23**
	RBOT	6.64	2.43	7.08	**10.86**	11.67
House	Proposed	**15.19**	**13.70**	**7.78**	**16.12**	**14.67**
	RBOT	11.36	2.05	3.40	11.02	7.82
Ironman	Proposed	**16.34**	**10.86**	**14.75**	**14.92**	**16.85**
	RBOT	6.52	5.45	5.78	5.41	8.81
Average	Proposed	**13.43**	**11.44**	**12.06**	**13.74**	**14.58**
	RBOT	8.17	3.31	5.42	9.10	9.94

To evaluate how the segmentation manages motion blur, we have performed a comparison between segmented pixels and ground truth masks using all Level 5 motion (fastest) OPT sequences. Results in Table 4 show that our approach successfully handles blur, overcoming the blurred-edges problem of edge-based techniques by using the pigmented information.

Table 4: Average results for motion blur segmentation test.

Measure	Chest	House	Ironman
False Negatives (%)	3.50	1.98	6.82
True Positives (%)	98.75	96.99	96.74

We also performed combined tests to understand the impact of dynamic radii. The test compared variations on search range (fixed) to Eq.(1) in the Scale Changes (zoom) scenes of the OPT dataset and the Cat scene from the RBOT dataset. Table 5 shows dynamic radius is preferable for generality purposes.

Table 5: Results for different search ranges (in px).

Object	AUC or RBOT Score				
	$r = r_0$	$r = r_1/4$	$r = r_1/2$	$r = r_1$	Eq.(1)
Chest	13.65	13.71	**13.90**	**13.90**	13.69
House	2.48	7.72	7.81	**8.15**	7.78
Ironman	14.03	**15.63**	11.59	5.92	14.75
Average	10.05	**12.35**	11.10	9.32	12.07
Cat	57.90	67.30	94.10	88.70	**97.20**

Next, we use the textureless pigmented objects from RBOT's proposed dataset together with its proposed metric (Tjaden et al., 2018) on the scene *regular* to evaluate robustness to similar-color backgrounds by varying σ tolerance (see Table 6).

Numbers for the object Cat show that the proposed work excels in handling similar-color background when there's little uncertainty, even surpassing RBOT's high score, but suffers from background matching otherwise. Such low uncertainty values are only possible because the materials are very accurate to the objects, which means our technique requires high-quality 3D model materials to perform well un-

Table 6: Evaluation on varying σ values. Percentages S, TP, and FN stand respectively for score (Tjaden et al., 2018), and averages of true positives and false negatives (in %) for the segmented pixels with respect to the ground truth segmentation masks.

Object	**Proposed** (varying σ values)									**RBOT**
	0.10			0.05			0.01			
	S	TP	FN	S	TP	FN	S	TP	FN	S
Ape	36.7	82.6	9.5	**48.4**	92.5	**7.6**	47.4	**98.1**	33.5	85.2
Duck	43.8	96.2	**5.5**	**52.2**	99.7	6.1	44.9	**100.0**	21.6	87.1
Cat	13.5	43.7	18.8	50.4	81.2	**2.7**	**97.2**	**99.5**	7.1	95.5
Squirrel	25.4	67.8	18.5	37.5	77.2	**15.5**	**70.5**	**97.8**	34.2	99.2

der similar-color backgrounds. Ape and Duck were precisely segmented as well, but their quick-rotating, simple silhouettes created inconsistencies in the estimated poses. Squirrel was not properly segmented even at low uncertainty due to large portions of *same-color* background, which is a limitation of our segmentation approach (different from *similar-color*).

We also evaluated the different challenges of the RBOT dataset for the Cat object, still using RBOT's metric. Results from Table 7 show our technique handles the challenges comparably to RBOT.

Table 7: Comparison of RBOT scores per challenge for the Cat object (higher is better).

Method	Regular	Dynamic Light	Occlusion	Average
Proposed	**97.2**	**97.0**	87.0	**93.7**
RBOT	95.5	95.2	**90.1**	93.6

One thing to note is that RBOT's metric is very strict, resetting the tracker when it is not necessarily lost. Thus, to show how the proposed technique is able to recover from partial tracking losses on its own, we have used a no-reset methodology (Garon and Lalonde, 2017) on the RBOT dataset, and plotted the translation and rotation errors separately for the regular and occlusion scenes, without ever resetting the trackers since the first frame. Results from Fig. 7 show that our work is always able to recover the object's position (translation), whereas RBOT is unable to recover since the first loss.

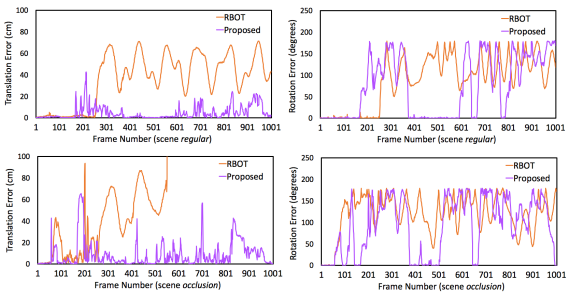

Figure 7: Recovery from partial loss results.

Because detection module calls are usually computationally expensive, this reduced need is highly desirable. In addition, this adds flexibility to minor losses right from the start or the tracking process, meaning there's less requirement for a robust detection module. This fits well with our proposed low hardware requirement application scenario, especially considering that popular mobile trackers can be manually-aligned by a user[6] and that state-of-the-art detectors use deep learning and powerful GPUs.

6.2 Imprecise Initialization

To further understand how robust our technique is to an imprecise previous pose, we have conducted the very extensive Imprecise Initialization test (Garon and Lalonde, 2017). For this test, 10 frames were selected at random from the sequence Slow of the 3DPO dataset, which has 1155 frames at 30 Hz. This sequence (unlike OPT ones) has a lot of clutter and background gradients. Unlike the RBOT dataset's, the sequence is also not synthetic, containing motion blur (an important aspect for temporally-consistent real-time tracking). It consists of smooth camera movements around a still Green Bunny, with comprehensive camera rotations and translations while moving closer and farther away from the object. Ground truth was obtained with the help of ArUco markers.

Initial pose rotation and translation perturbations are measured separately. Rotation perturbations range from 5 to 60^o and are incremented 5^o at a time. Translation perturbations range from 10 to 130 mm in 10 mm increments. For each increment, 40 random samples are taken. Before computing the error, the tracker is called 15 times for each frame to take the temporal factor into consideration. Mean and standard deviation for the L2 norms of the errors (with respect to the ground truth) of each of the 10 sample for each of the perturbations are plotted in Fig. 8.

The proposed technique has achieved consistent translation robustness for up to 60 mm, which corresponds to roughly 30% of the model's size. For rotation, the technique achieved constantly reliable results for up to 15^o perturbation. The deep learning work that first proposed this methodology has also performed well in this test (Garon and Lalonde, 2017) (albeit using a different RGB-D sequence and object). It achieved robustness levels of 100 mm and 30^o. In their case, though, all perturbation averages (even the smallest ones) had at least 8 mm and 8^o of error. This is a good illustration of the trade-offs between geometric and machine learning methods. The similarity between these very different approaches is that

[6]Vuforia Model Targets

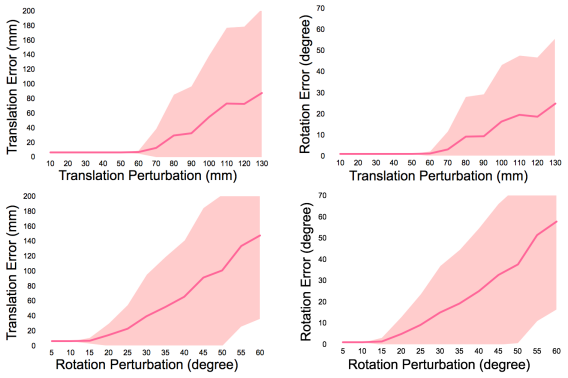

Figure 8: Initial perturbation test. Lines represent the average error. Margins represent the standard deviation.

both are supplied with some sort of previous knowledge regarding the model's color and texture. Model-based techniques do not tend to directly assess this issue, simply falling back to a costly detection module. Since real-world applications usually can not count with a ground truth value, robustness to detection error is a relevant factor and should not be overlooked. This serves as argument for using texture information alongside geometry on model-based approaches.

6.3 Performance and Scalability

The first frame from the RBOT dataset sequence *regular* and the object Cat (with 2500 vertices) were used for this test. More object instances were added on top of the same object to simulate a multi-object scenario. The proposed work's memory requirement was orders of magnitude smaller than the state of the art's, at $\sim$ 10 MB per object tracked simultaneously, on top of a baseline value $\sim$ 10 MB. RBOT's memory consumption was $\sim$ 4.3 GB per object, which makes multi-object impractical for most low-end devices. In terms of runtime, RBOT scaled linearly with $\sim$ 70 ms per new object. The proposed work had an initial runtime of $\sim$ 30 ms, with approximately $\sim$ 3 ms per new object until a limit of 3 objects (amount of threads able to run in parallel efficiently on our dual-core CPU). Afterwards, our work's runtime scaled linearly with $\sim$ 30 ms per new object.

The second test had a similar setup but used only one object at a time. Cat objects with different simplification levels (amount of vertices) were used, as follows: 2500, 5000, 7500, 10000, 12500. RBOT scaled at $\sim$ 4.3 GB of RAM per new 2500 vertices, whereas the proposed work showed no significant difference, staying at about 20 MB. In terms of runtime, RBOT scaled $\sim$ 10 ms per 2500 vertices from a baseline of $\sim$ 70 ms. The proposed work scaled about 2 ms per 2500 vertices, from a baseline of $\sim$ 30 ms. Our single-object mobile port ran at $\sim$ 50 ms on a Samsung Galaxy S10 phone.

Our work averaged roughly 2.5 ms per pipeline iteration, consisting of: encoding, segmentation, matching, and pose estimation. Preprocessing averaged 112 ms per 100 triangles with diffuse texture and 3 ms for those without.

7 CONCLUSION

We have presented a novel edge-based 6DoF tracking technique focused on textureless, pigmented objects. Our work, for the intended scope of objects, achieves comparable results to the present state of the art while surpassing it in relevant aspects. Our technique particularly excels in challenges like illumination changes, motion, and recovering from imprecise poses. It is also single-core, RGB-only, and much more lightweight, thus more well suited for less-powerful hardware. We have also proposed a new way to avoid fixed search ranges, an efficient silhouette encoding technique, a study on using material information for model-based 6DoF object tracking, and a new segmentation step which handles problems such as background gradients and gradients lost to motion blur. Finally, we have made available sequences focused on 3D printed textureless objects that we hope can enrich further research.

As future work, our technique's main limitations have proven to be symmetry, which can be aided by photometric constraints (Zhong and Zhang, 2019); and imprecise textures along with similar or same-color backgrounds, which can be tackled by using temporal foreground-background segmentation strategies as performed by region-based approaches.

ACKNOWLEDGEMENTS

This research was partially funded by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) (process 425401/2018-9) and HP Brasil Indústria e Comércio de Equipamentos Eletrônicos Ltda. with resources from the fiscal benefit of the Brazilian Law n^o 8.248, from 1991 (Brazilian Informatics Law). The authors thank Voxar Labs members Lucas Maggi, Rafael Roberto, Gutenberg Barros, Kelvin Cunha, Heitor Felix, and Thiago Souza for helpful discussions. Tsang Ing Ren and Ricardo Barioni are also thanked for helpful discussions as well as for proofreading this manuscript. Finally, Henning Tjaden, Bogdan Bugaev, and Po-Chen Wu

are thanked for helping the authors better understand RBOT and OPT.

REFERENCES

Bugaev, B., Kryshchenko, A., and Belov, R. (2018). Combining 3d model contour energy and keypoints for object tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 53–69. 2, 7

Drummond, T. and Cipolla, R. (2002). Real-time visual tracking of complex structures. *IEEE Transactions on pattern analysis and machine intelligence*, 24(7):932–946. 5

Garon, M. and Lalonde, J.-F. (2017). Deep 6-dof tracking. *IEEE transactions on visualization and computer graphics*, 23(11):2410–2418. 2, 8

Han, P. and Zhao, G. (2019). A review of edge-based 3d tracking of rigid objects. *Virtual Reality & Intelligent Hardware*, 1(6):580–596. 2

Harris, C. and Stennett, C. (1990). Rapid-a video rate object tracker. In *BMVC*, pages 1–6. 2

Lima, J. P., Teichrieb, V., Kelner, J., and Lindeman, R. W. (2009). Standalone edge-based markerless tracking of fully 3-dimensional objects for handheld augmented reality. In *Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology*, pages 139–142. 3

Lima, J. P. S. d. M. (2014). Object detection and pose estimation from rectification of natural features using consumer rgb-d sensors. 4

Mur-Artal, R. and Tardós, J. D. (2017). Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262. 2

Prisacariu, V. A. and Reid, I. D. (2012). Pwp3d: Real-time segmentation and tracking of 3d objects. *International journal of computer vision*, 98(3):335–354. 2

Seo, B.-K., Park, J., Park, H., and Park, J.-I. (2013). Real-time visual tracking of less textured three-dimensional objects on mobile platforms. *Optical Engineering*, 51(12):127202. 2, 5

Stoiber, M., Pfanne, M., Strobl, K. H., Triebel, R., and Albu-Schaeffer, A. (2020). A sparse gaussian approach to region-based 6dof object tracking. In *Proceedings of the Asian Conference on Computer Vision*. 2

Suzuki and Abe (1985). Topological structural analysis of digitized binary images by border following. *Computer vision, graphics, and image processing*, 30(1):32–46. 3, 5

Tan, D. J., Navab, N., and Tombari, F. (2017). Looking beyond the simple scenarios: Combining learners and optimizers in 3d temporal tracking. *IEEE transactions on visualization and computer graphics*, 23(11):2399–2409. 3

Tjaden, H., Schwanecke, U., Schömer, E., and Cremers, D. (2018). A region-based gauss-newton approach to real-time monocular multiple object tracking. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1797–1812. 2, 7, 8, 11

Valença, L. (2020). Real-time 6dof tracking of rigid 3d objects using a monocular rgb camera. Bachelor's thesis, Universidade Federal de Pernambuco. 2

Wang, C. and Qian, X. (2018). Heaviside projection–based aggregation in stress-constrained topology optimization. *International Journal for Numerical Methods in Engineering*, 115(7):849–871. 5

Wang, G., Wang, B., Zhong, F., Qin, X., and Chen, B. (2015). Global optimal searching for textureless 3d object tracking. *The Visual Computer*, 31(6-8):979–988. 2

Wu, P.-C., Lee, Y.-Y., Tseng, H.-Y., Ho, H.-I., Yang, M.-H., and Chien, S.-Y. (2017). [poster] a benchmark dataset for 6dof object pose tracking. In *2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR-Adjunct)*, pages 186–191. IEEE. 2, 7

Zhong, L. and Zhang, L. (2019). A robust monocular 3d object tracking method combining statistical and photometric constraints. *International Journal of Computer Vision*, 127(8):973–992. 2, 9

APPENDIX

Depth Testing

Because we're only interested in the outermost 2D silhouette edges, overlap is infrequent. Objects with rounded edges do not contain 2D silhouette edges overlapping. Objects with straight-angled edges (such as boxes) avoid significant silhouette overlap due to perspective projection. To validate this, we implemented a simple CPU-based z-buffer. As expected, for rounded-edge objects it made little to no difference, and for the straight-angled ones in scenes where the edges were overlapping results were slightly better. Yet, to our surprise, depth-testing made results much worse when dealing with rotations. A possible explanation is that by allowing some edge overlap, we can better segment colors that are not directly visible, but that can appear with a small rotation. Thus, as not including depth testing made our work both faster and more precise, we have opted to not utilize it.

Saturation Threshold

By definition, HSV space only has overlap in hue values when there is zero pigmentation present, in which case all hue values can lead to the same color. That is not true in the real world, though, where illumination, materials, camera sensors, and color-conversion algorithms play a role in the final color. This concept can

be better illustrated via a *print-scan* attack. Except, instead of a scanner, we have used a webcam and a smartphone (see Fig. 9).

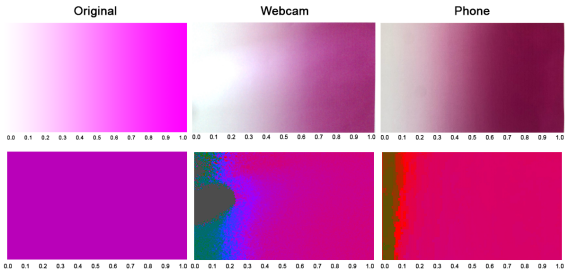

Figure 9: Illustration of hue unreliability as saturation decreases. Top row shows the full RGB images, bottom row the isolated hue channel.

Publicly Available Implementations

As far as we are aware (as of the time of the writing of this paper), the best publicly available real-time RGB 6DoF trackers, which are commonly used for comparisons, are PWP3D, GOS, and RBOT.

PWP3D is nearly 10 years old and has since inspired and been surpassed by newer works, especially RBOT, which has compared itself to it extensively. Thus, adding PWP3D to the comparison would not accurately represent the current state of the art.

As for GOS, it is also fairly outdated, having inspired newer techniques. Still, as far as we know, no newer techniques built on top of it have a publicly available implementation. In our tests, GOS's results averaged below 35% on the RBOT dataset for all objects in every scene (with the reset metric). GOS stayed below 1% without a reset. It was completely unable to recover and extremely sensitive to motion, which caused it to fail almost immediately in all 3DPO sequences when the objects or the camera were moved. Thus, we felt it was out of place in the comparison.

Next, we are going to explain why our results using the publicly available implementation of RBOT on the OPT dataset are lower than what is reported in RBOT's paper (Tjaden et al., 2018). First, it is important to note that we did manage to reproduce RBOT's paper's results for the RBOT dataset. We also confirmed our AUC metric to be correct by reproducing the OPT results of other works which used the publicly available implementation of GOS. Upon contacting the authors of RBOT, we were informed our AUC metric seemed correct, but that OPT objects needed some resampling to work with the publicly available implementation of RBOT. Unfortunately, the authors did not have access to the modified objects anymore. In our attempts, we were unable to make the publicly available implementation of RBOT reach the OPT scores reported in the paper, so we decided to use the best results we were able to get. There was no way around this issue, as we needed to run the technique to build the AUC curves and conduct challenge-specific tests. Still, we want to make it clear that RBOT's publicly available implementation does work very robustly (even in our highly challenging 3DPO sequences). It is also important to note that these OPT results are still comparable to the best publicly available real-time monocular 6DoF trackers.